2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Advances in RF Transceiver SoC Verification:

A Walk-Through over a 2.4 GHz Multi-Modal Integrated Transceiver Verification Cycle

Charul Agrawal, Ashwin Vijayan, Jakub Dudek

Analog Devices, Inc.

ANALOG DEVICES

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Agenda : Walkthrough

- Challenges in Radio SoC Verification
- Advances for Block Level Verification
- Advances for System Level Verification
- Advances for Firmware Development
- Conclusion and Silicon Results

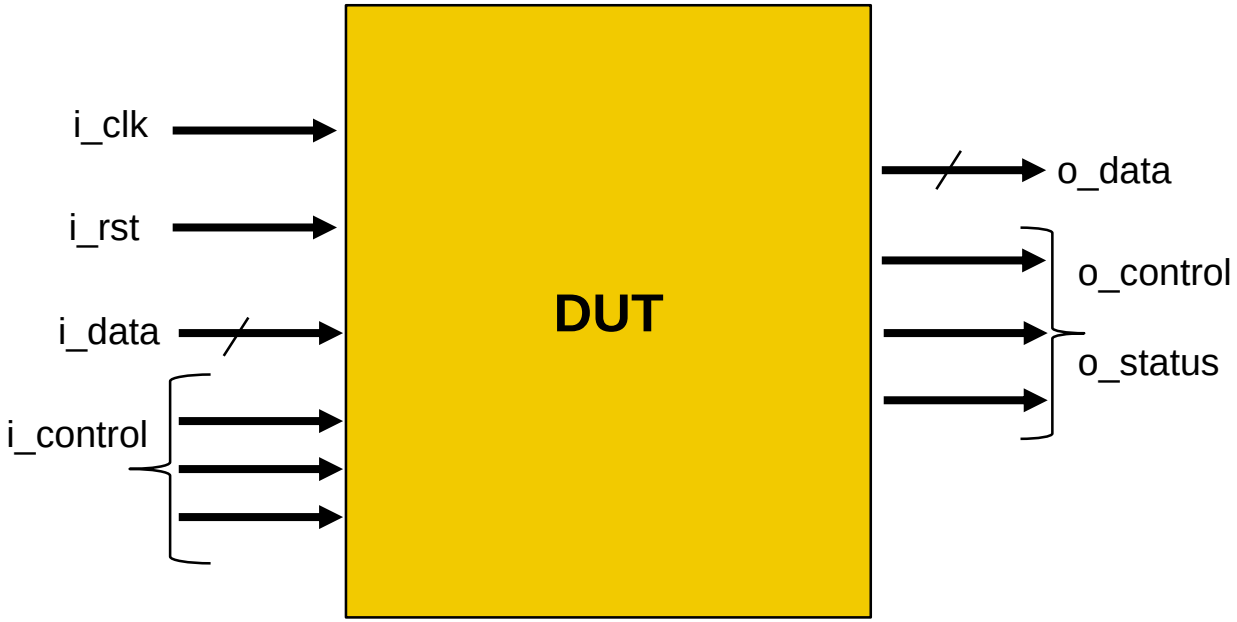
Challenges in Radio Verification

- Complex SoCs
- Mixed Signal Control Loops
- Slow CoSim
- Stimuli Dependent Configuration
- Blockers
- Dependence on MATLAB
- Delayed Firmware Development

Block Level Advances

Signal processing block

- Custom interfaces for each block
- Few static control signals
- Data signal which is transformed



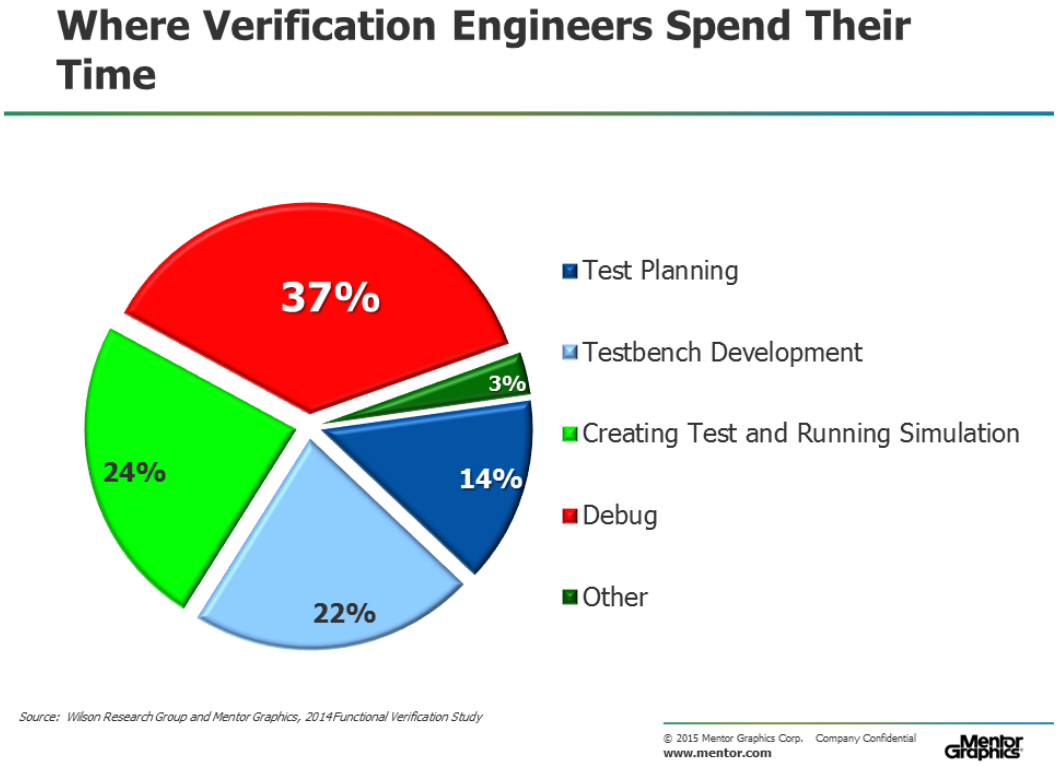
Block Level Verification : Traditional Approach

Custom Env.

- Low portability
- Less re-use
- Time consuming

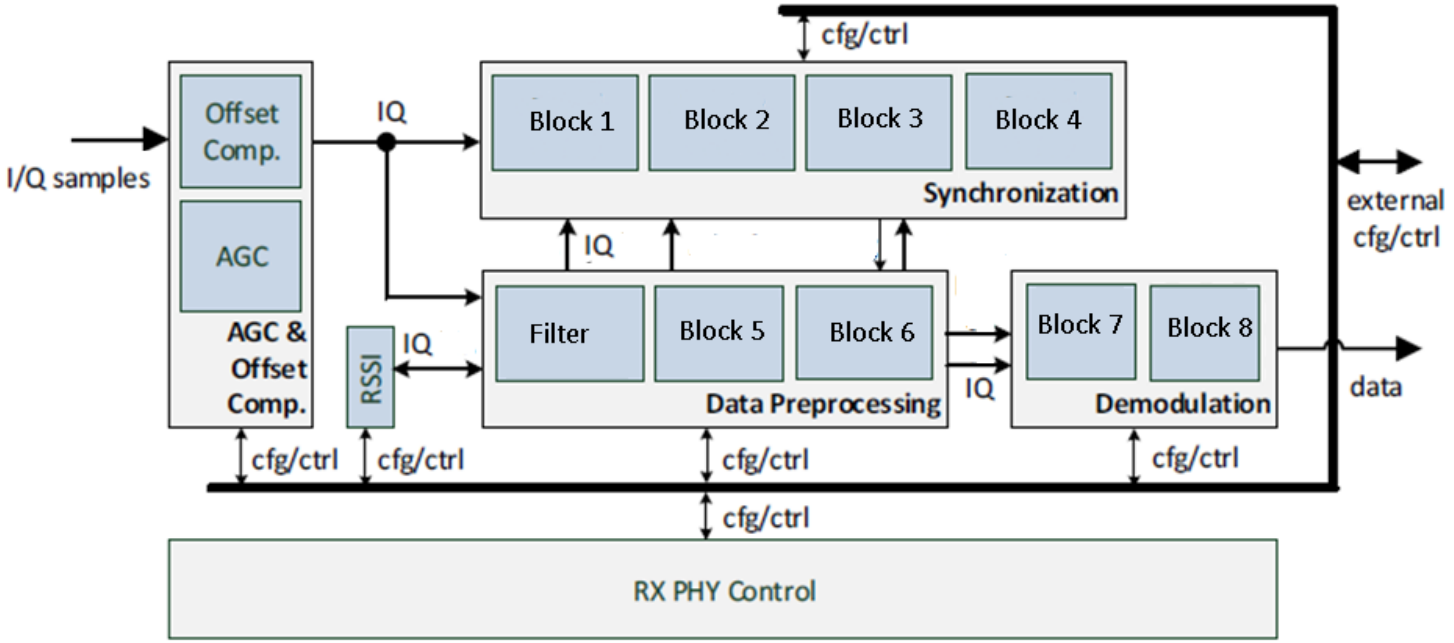
File Based

- MATLAB etc.
- Limited stimuli
- Simulation time



Block Level Verification : Traditional Approach

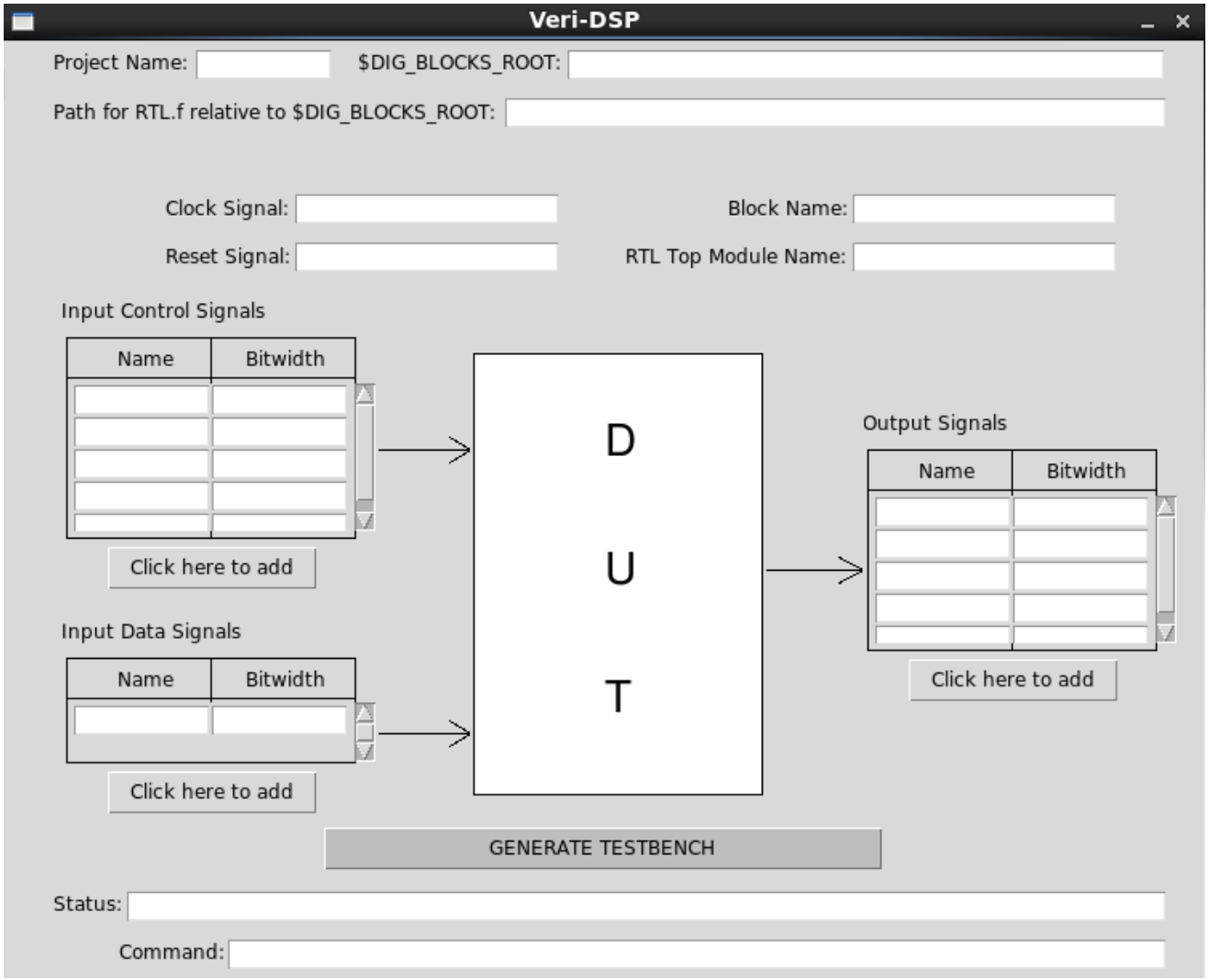
- Hugely challenging due to numerous signal processing designs both in Tx and Rx paths



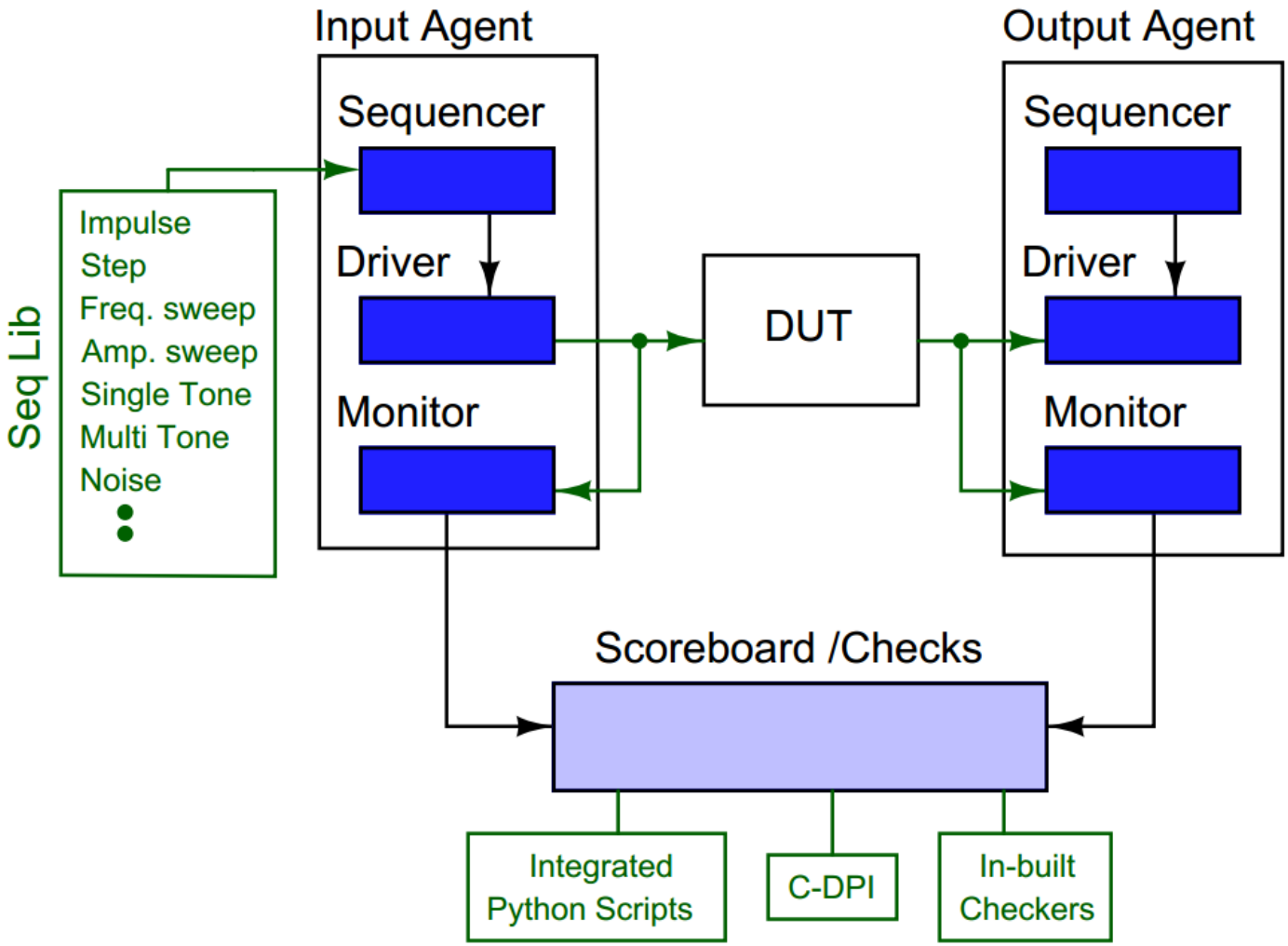
Automatic TB generation

- **Automating the testbench generation** with in-built tests, stimuli, checkers that can be run out-of-the box, can substantially reduce the overall verification time.
  - Aids design engineers in early debug
  - Increases productivity across different projects

Ease of use


Veri-DSP
Project Name:
$DIG_BLOCKS_ROOT:
Path for RTL.f relative to $DIG_BLOCKS_ROOT:
Clock Signal:
Block Name:
Reset Signal:
RTL Top Module Name:
Input Control Signals
Name
Bitwidth
Click here to add
Input Data Signals
Name
Bitwidth
Click here to add
D
U
T
Output Signals
Name
Bitwidth
Click here to add
GENERATE TESTBENCH
Status:
Command:

Generated TB Architecture



Substantial time savings

- Verified blocks such as Digital Channel Filter, Interpolator, RSSI, Timing Sync., Frame Sync., Demodulator etc.

- Block-level DV for a 31-tap, fully programmable FIR filter completed in 4 weeks by a beginner.

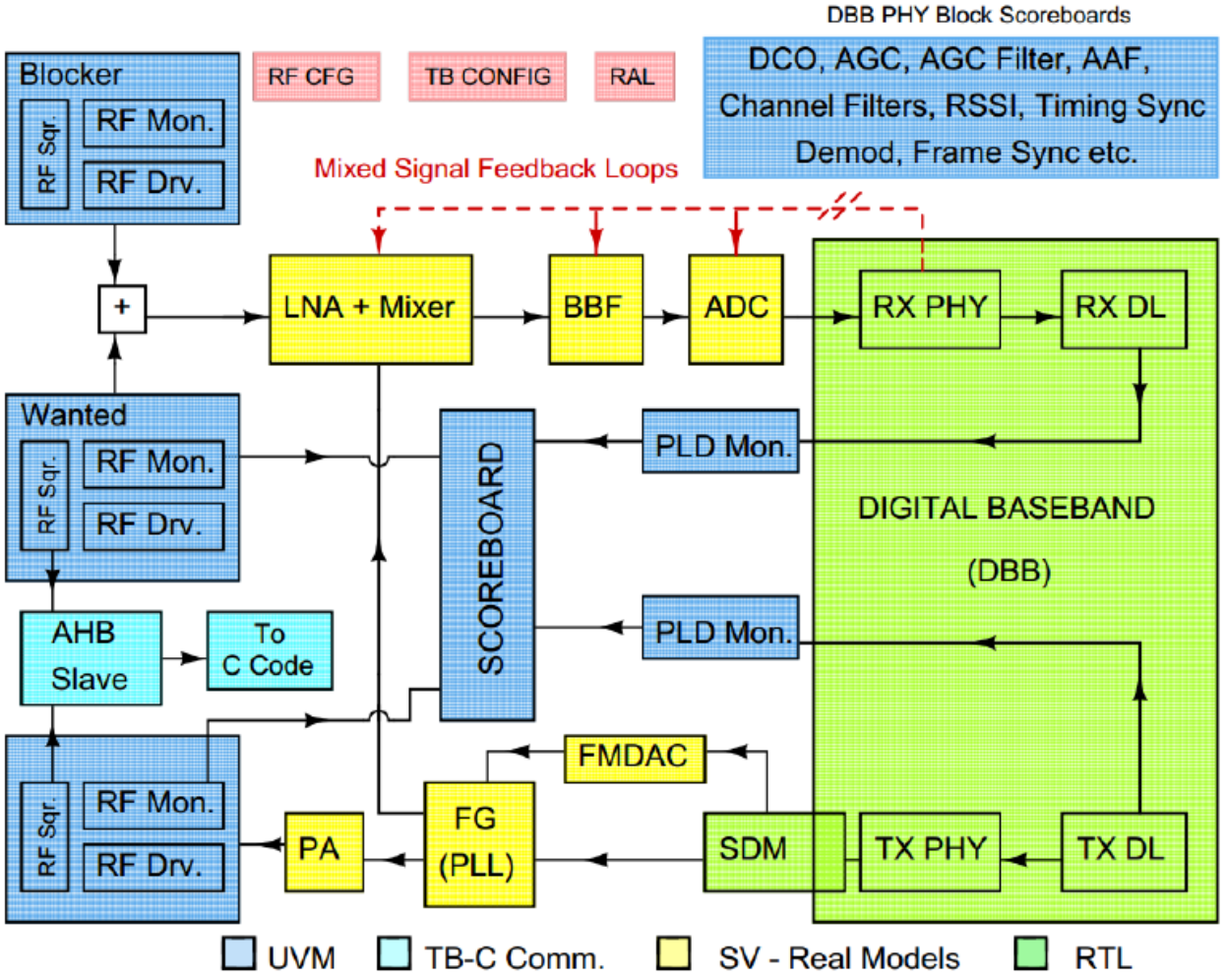
- Regular approach required additional **2-3 weeks** per block ➡ around **20-25 weeks** for entire DBB

System Level Advances

System Level Advances

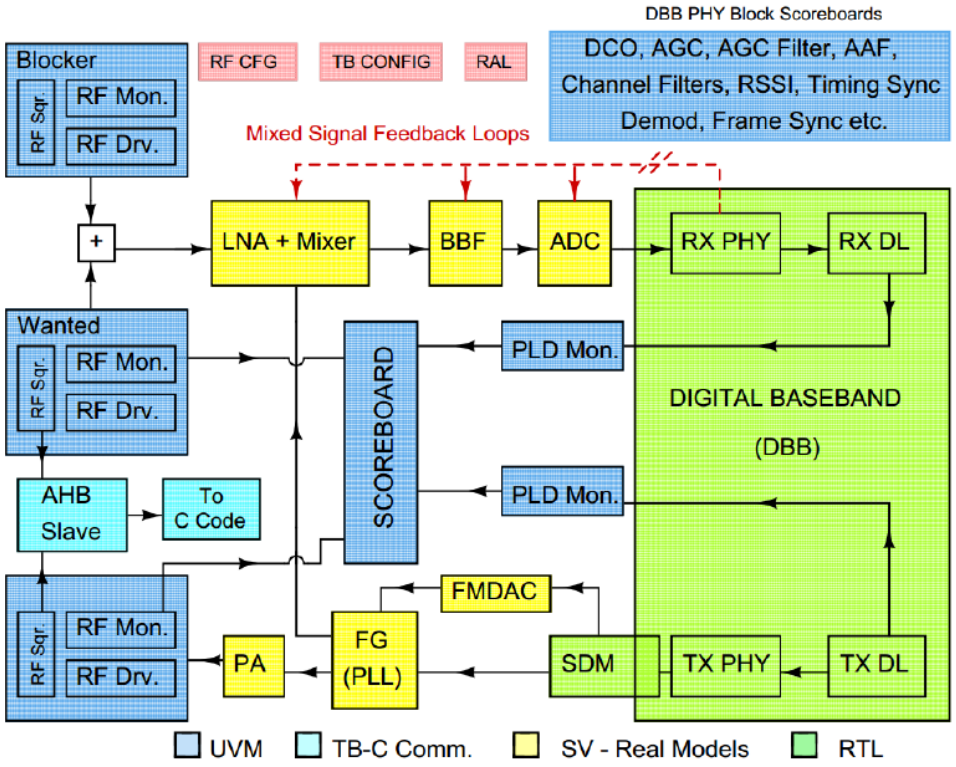
- RNM models for analog components of Radio:
  - LNA, PA, PLL, BBF, IF Mixer, etc.
- Digital-centric DV environment
- Communication between test and C code

Testbench Architecture



RF Agents

- LNA : Active
- PA : Passive



- Monitor
  - Monitors SOF, EOF, Data
  - Reconstruct Packets

Transaction

- About 30 parameters randomized

Randomizable Paramaters		
Address	channel	header size
preamble	protocol	pld min
postamble	MI	pld max
midamble	BITRATE	pld crc size
rampup	direction	crc init
rampdn	f err	crc poly
dBm	packet type	hcrc init
whitening	pkt_q	hcrc_poly

FC Report

Average Grade	Covered Grade	Goal	Weight	Uncovered Bins	Excluded Bins	Total Bins	Item	Name
100.00%	100.00% (6/6)	100%	1	0	0	6	CoverPoint	rx_cg.rx_pkt_q_size
100.00%	100.00% (3/3)	100%	1	0	0	3	CoverPoint	rx_cg.rx_protocol
100.00%	100.00% (9/9)	100%	1	0	0	9	CoverPoint	rx_cg.rx_preamble_size
100.00%	100.00% (3/3)	100%	1	0	0	3	CoverPoint	rx_cg.rx_postamble_size
100.00%	100.00% (3/3)	100%	1	0	0	3	CoverPoint	rx_cg.rx_midamble_size
100.00%	100.00% (7/7)	100%	1	0	0	7	CoverPoint	rx_cg.rx_channel
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	rx_cg.rx_header_size
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	rx_cg.rx_pld_crc_size
100.00%	100.00% (2/2)	100%	1	0	0	2	CoverPoint	rx_cg.rx_whiten
100.00%	100.00% (97/97)	100%	1	0	0	97	CoverPoint	rx_cg.rx_dbm
100.00%	100.00% (10/10)	100%	1	0	0	10	CoverPoint	rx_cg.rx_f_err
100.00%	100.00% (23/23)	100%	1	0	0	23	Cross	rx_cg.__rx_protocol_X_rx_preamble_size
100.00%	100.00% (18/18)	100%	1	0	0	18	Cross	rx_cg.__rx_protocol_X_rx_channel
100.00%	100.00% (6/6)	100%	1	0	0	6	Cross	rx_cg.__rx_protocol_X_rx_whiten
100.00%	100.00% (70/70)	100%	1	0	0	70	Cross	rx_cg.__rx_f_err_X_rx_channel
100.00%	100.00% (6/6)	100%	1	0	0	6	Cross	rx_cg.__rx_protocol_X_rx_midamble_size
100.00%	100.00% (6/6)	100%	1	0	0	6	Cross	rx_cg.__rx_protocol_X_rx_postamble_size
100.00%	100.00% (3/3)	100%	1	0	0	3	Cross	rx_cg.__rx_protocol_X_rx_header_size
100.00%	100.00% (4/4)	100%	1	0	0	4	Cross	rx_cg.__rx_protocol_X_rx_pld_crc_size
100.00%	100.00% (237/237)	100%	1	0	0	237	Cross	rx_cg.__rx_protocol_X_rx_dbm
100.00%	100.00% (24/24)	100%	1	0	0	24	Cross	rx_cg.__rx_protocol_X_rx_f_err
100.00%	100.00% (13/13)	100%	1	0	0	13	Cross	rx_cg.__rx_protocol_X_rx_pkt_q_size

Transaction

- Burst mode supported using queue of packets
  - Constrained packet-to-packet randomization

```
for(int i = 0; i<pkt_q.size(); i++) begin
pkt_q[i] = packet::type_id::create("", null);
                                                    void'(pkt_q[i].
randomize() with {has_hcrc == |h_crc_poly; header.size ==
header_size; payload.size inside {[pld_min:pld_max]};
packet_type == local::packet_type;});
 end
```

Blocker generation

To verify interference performance

- RF agent used for blocker generation
- At LNA port
- Blockers with different channel spacing could be generated

Scoreboard checks

- Comparing PA and LNA packets

```
foreach(lna_q[i]) begin
  // ..............Some code ............
  if (!lna_q[i].compare(rxpld_q[i])) begin
    `uvm_error(get_type_name(), $sformatf(
      "Miscompare at Payload monitor, packet %0d", i))
     foreach(lna_q[i].payload[j]) begin
       if(lna_q[i].payload[j]!=rxpld_q[i].payload[j])
         `uvm_info(get_type_name(), $sformatf(
                    "[%d] %x %x", j, lna_q[i].payload[j],
                    rxpld_q[i].payload[j]), UVM_LOW);
    end
    break;
  end
end
```

TB-C Communication

Motivation

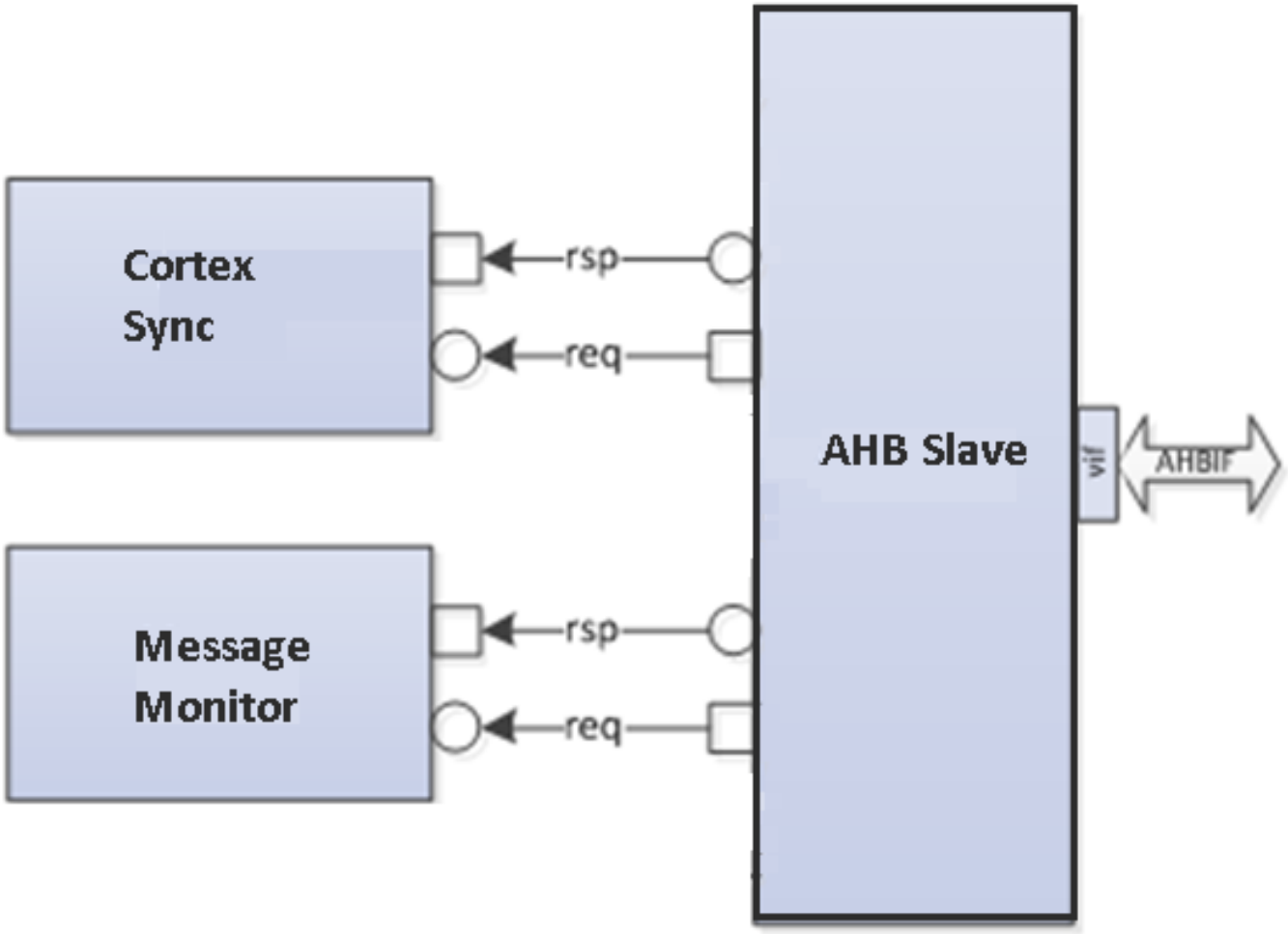
- Rx DBB to be configured according to transmission parameters
- Obstruction in debug with non-reproducible rand() used in C code for processor
- Handshaking between Test and C code

TB-C Communication

Method

- AHB slave in TB for transactions at virtual address space
- Slave has ability to toggle response and read data lines : Normal memory for processor

TB-C Communication



TB-C Communication

- Each block agent can register a pair of TLM ports by passing in the wanted index of the associative array

- Each component can define its own put() and get() function

TB-C Communication

```
function void register(int id);
  if(this.reqs.exists(id))
   `uvm_warning("INIT", $sformatf("id %d already registered",
              id))
  else
  begin
    uvm_blocking_put_port#(int)
              req = new($sformatf("req%0d", id), this);
    uvm_blocking_get_port#(int)
              rsp = new ($sformatf ("rsp %0d",id), this);
    this.reqs[id] = req;
    this.rsps[id] = rsp;
  end
endfunction
```

TB-C Communication

- Passing transmission parameters
  - Configuration structure was declared
  - Populated by calling the get_rx_config() and get_tx_config() routines

```
tcfg = get_rx_config();

rx_cfg.agc_cfg = agc_cfg_btle_init();
rx_cfg.framesync_cfg = framesync_cfg_init(&tcfg);
rx_cfg.cfo_cfg = cfo_cfg_init();

…

Idle_To_Phy_On(&rx_cfg);
```

Handshaking - Code

sync2tb and wait4tb

```
task run_phase(uvm_phase phase);
rf_seq_item transmission;
rf_sequencer sqr =
env.lna_agent.seqr;
…
void’(transmission.randomize())
with {…};
proc.sync.send_sync(`RX_SYNC);
…
sqr.current_item = transmission;
// Wait for C code to get info.
sqr.proc_sync.wait_for_sync(`RX_S
YNC);
sqr.execute_item(transmission);
proc_sync.send_sync(`RX_SYNC);
…
```

```
int main(void) {

wait4tb(`RX_SYNC);
tcfg=get_rx_config();
Ready_To_Idle();
Idle_To_Phy_On();
…
// Resume seq in TB
sync2tb(`RX_SYNC);
…
// Wait for run phase to
complete
wait4tb(`RX_SYNC);

return EXIT_SUCCESS;
}
```

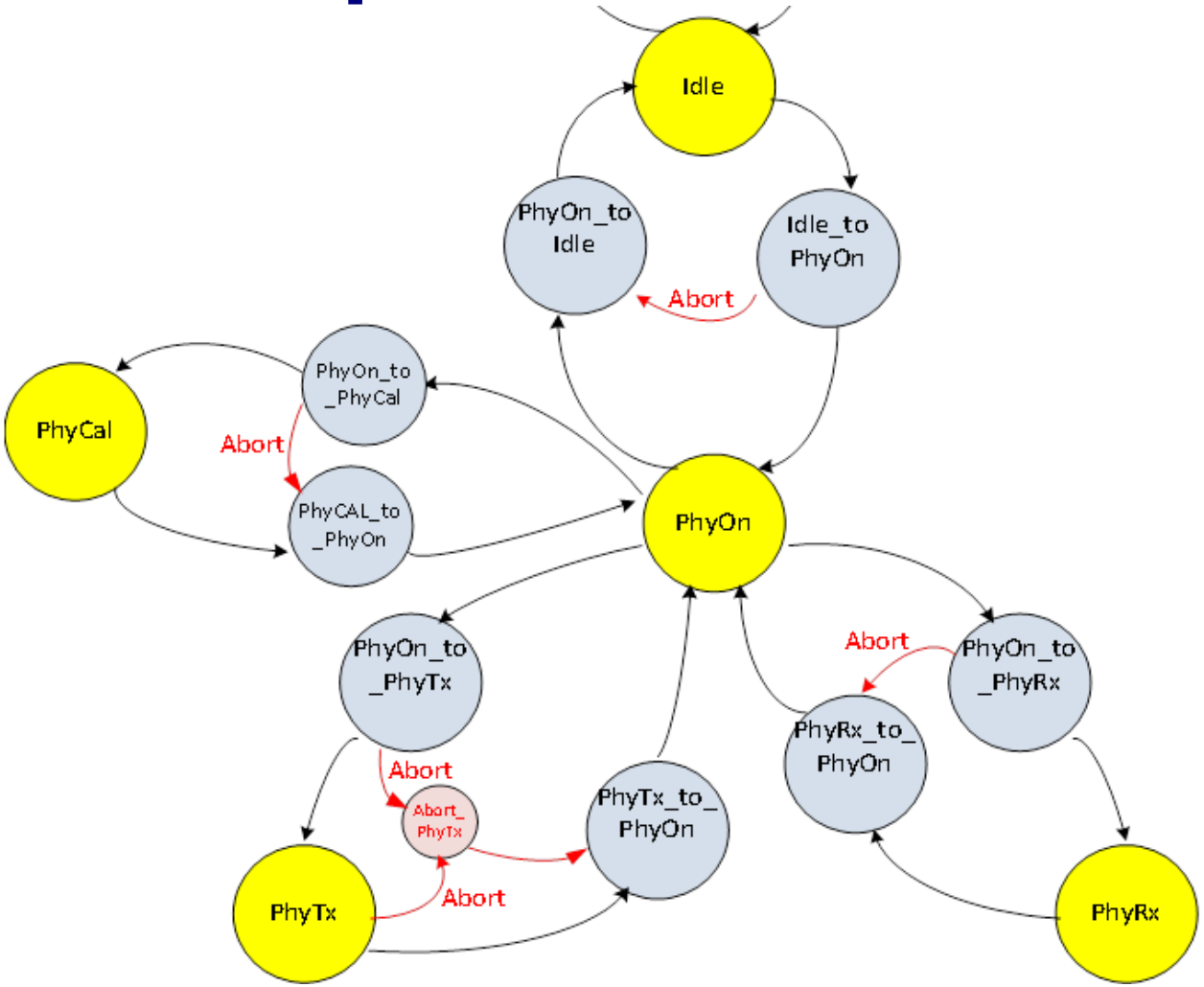
Advances in Firmware development

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Pseudo FW approach

- FW development needs re-runs : Time consuming with Cosim
- Development on FPGA starts late in project cycle
- Faster RNM based simulation environment enabled early bring up
- FW alike code used with same state and function names
- FW code tested on DV environment and FPGA

FW states and functions : Example



2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Conclusion

- Fast block TB bring up, even by beginners
- No bugs found for digital blocks in CoSim
  - Digital stable even before cosim started
- Exhaustive randomization achieved for transmission parameters
- Fast firmware development
- Real use case simulation with pseudo-FW approach

2017 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Reusable Environment

- Extensible RNM models
- Updatable RF agents for new modulation schemes

Silicon

- Fast silicon bring up
- **No major bug found in silicon** evaluation so far
- **Silicon shipped to customer 1.5 weeks ahead of schedule**
  - **Within one year of project launch**

Thank you